Molecule

Paper by: Sébastien Bocq & Koen Daenen

Presentation by: Ahmad Salim Al-Sibahi
Disposition

• Introduction
• Molecule Framework
  – Processes
  – Channels
  – Termination
  – Scheduling
• Evaluation
• Conclusions
Goal of the project

INTRODUCTION
Objective

A high-performance library-based JVM framework enabling composition of process networks
Motivation

Existing Frameworks:
1. Callback methods error prone

2. Compiler-based transformations hard to maintain

3. Exceptional conditions require explicit propagation
Key Achievements

- Pure Scala-based library
- Compositional I/O using data streams
- Encapsulated system-wide resource and network management
- High-performance parallelism using Flow-parallel scheduling
High-performance safe CSP’s

MOLECULE
Processes

• Definition
  – Isolated concurrent sequence of computation
  – Lack of shared (writeable) data

• Design

<table>
<thead>
<tr>
<th>Callback Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>new IO[Baking]((t,k) =&gt; kneadAsync(dough =&gt; t.submit(bakeAsync(dough, bread =&gt; t.submit(eatAsync(bread, result =&gt; t.submit(k(result))))))))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compositional Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>knead() &gt;&gt; \</td>
</tr>
<tr>
<td>dough =&gt; bake(dough) &gt;&gt; \</td>
</tr>
<tr>
<td>bread =&gt; eat(bread)</td>
</tr>
</tbody>
</table>
Communication

Process Channels

Stream Channels

System-Level Channels
Termination

- Thread Contexts
  - Termination
  - Handling Exceptions
  - Channel Management

- Message Poisoning
Scheduling

• Naïve Scheduling

• Flow-parallel Scheduling
  – Thread reuse
  – Continuation-based suspension
  – Complexity and Segment-based Thresholds
Benchmarking

EVALUATION
Thread ring
Genuine Prime Sieve

Notice! Logarithmic Scale
Chameneos Redux
CONCLUSIONS

Criticism and Future work
Criticism

• The Benchmarking Game
  – Languages
    • Akka/Scalaz/Lift actors
    • Erjang
    • Java
  – Methodology
    • Hotspot Internals: Micro-Benchmarks

• Availability
Future work

• Support plain Java applications

• Examine performance benefits on real-life applications

• Generalize framework ideas to other languages
Commentaries and curiosities

QUESTIONS